Enqpy™ Performance Benchmark Report

Public Summary · Rev 5.0 · NQP LLC · June 2026

Proved permanent secrecy, with the speed to run it everywhere.

Summary

Enqpy™ is proved, and it's fast. On a level playing field — every cipher built as portable C with no hardware acceleration, the same message size, authentication included — Enqpy™ with Poly1305 runs roughly **2.5–2.8× faster than ChaCha20-Poly1305** and far ahead of portable AES-256-GCM. The reference passes 84/84 self-test assertions against the published test vectors. The throughput *ratio* measured within a single round-robin run is the stable, portable claim; absolute MB/s are core- and load-dependent, and a public harness (`aead_bench.c`) is provided so any reader can reproduce the result.

Method — a level playing field

- All ciphers run as self-contained portable C: no hardware dispatch, no intrinsics, no AES-NI.
- 256-bit keys throughout; an identical 59,189-byte message for every algorithm in every pass.
- Authentication included as deployed: Poly1305 for Enqpy™ and ChaCha20, GHASH for AES-256-GCM; AES-256-CTR is shown unauthenticated (noted as such).
- 10 round-robin warmup passes, then 20 timed round-robin iterations; best-of-20 and average reported.
- GCC 13.x, `-O3 -march=native -std=c11`; timing via `clock_gettime(CLOCK_MONOTONIC)` around the cipher call only.
- Every comparison primitive is verified against its official known-answer test before timing; round-trip decrypt is verified for all. Measurement does not proceed unless all checks pass.

Measurements were taken on a shared-tenant cloud core, where absolute throughput varies with co-tenant load (observed up to ~1.5× run-to-run). For that reason the ratio between ciphers within the same run — where every algorithm sees the same core state — is the claim; absolute figures are representative-with-range.

Software results

Algorithm (authentication included)	Best MB/s	Avg MB/s	vs Enqpy™
Enqpy™ (256-bit) + Poly1305	~900	~770	—
ChaCha20-Poly1305	~351	~310	~2.6× slower
AES-256-CTR (no authentication)	~185	~176	~4.9× slower
AES-256-GCM (portable GHASH)	~10.4	~10.1	~90× slower

Best-of-20 ranged 830–980 MB/s for Enqpy™ across runs on the shared core; the Enqpy-to-ChaCha20 ratio held at 2.5–2.8× across all runs and is the portable claim.

On hardware acceleration: the AES-256-GCM figure above is portable GHASH *without* PCLMUL. With PCLMUL/AES-NI on the same CPU, AES-256-GCM runs at several thousand MB/s, closing

most of the gap. Hardware acceleration is excluded here because Enqpy™ has no equivalent x86-64 hardware path — its acceleration story is the FPGA, summarized below.

Why it is fast

- **The core is cheap.** Keystream is generated by a single 4-bit add per nibble — branch-free and byte-aligned, with no S-box and no carry-chain arithmetic. The ~2 KB working set fits in L1 cache.
- **The MAC is the co-equal cost.** Authentication is Poly1305, keyed from the keystream — the same construction and the same cost ChaCha20-Poly1305 pays. Because the cipher is now so cheap, the combined authenticated-encryption time splits roughly evenly between cipher and MAC.

Per-record overhead

Algorithm	Nonce / OR	Auth tag	Total
Enqpy™ PDAF_SEC	32 B (OR register)	16 B (Poly1305)	48 B
ChaCha20-Poly1305	12 B (nonce)	16 B (Poly1305)	28 B
AES-256-GCM	12 B (IV)	16 B (GHASH)	28 B
AES-256-CTR	16 B (IV)	0 B (none)*	16 B*

*Ciphertext length equals plaintext length in every case. *AES-256-CTR production overhead with HMAC-SHA-256 is ~48 B (16 B IV + 32 B tag). Enqpy™'s larger nonce/OR field is the one place it trades a few bytes for its per-record keying discipline.*

Hardware potential (attested — not publicly reproducible)

FPGA results are internal synthesis and simulation figures on proprietary RTL (Artix-7 class, 100 MHz), available for evaluation under NDA and therefore **attested rather than publicly reproducible**. They are reported here for context only; software, hardware, and primitive figures are distinct metrics on distinct platforms and are not directly comparable.

- Current RTL: ~6,400 MB/s peak at 64 bytes/clock (~5,965 MB/s sustained), scaling to ~12,800 MB/s at 128 bytes/clock, with low incremental logic cost because the output bus widens combinationally.
- A Case-1 Enqpy™ hardware variant has been functionally simulated in GHDL (cycle-accurate functional checks pass, and the key-equivocation coset (≥2-bit floor) reproduces in RTL), but has not yet been synthesized.

Hardware engineering, RTL evaluation, and commercial deployment are handled by NQP LLC.

Reproducibility

Build and run the published authenticated-encryption harness:

```
gcc -O3 -march=native -std=c11 -D_POSIX_C_SOURCE=200809L aead_bench.c enqpy_reference.c -o aead_bench && ./aead_bench
```

All comparison ciphers (AES T-tables, ChaCha20, Poly1305, GHASH) are implemented inline and known-answer-verified before timing; the Enqpy™ primitive is linked from `enqpy_reference.c`. No third-party libraries.

Verification

Enqpy™ reference: 84/84 self-test assertions pass against the published test vectors, and all round-trip encrypt/decrypt tests pass.

© 2026 NQP LLC · Enqpy™ Performance Benchmark Report — Public Summary, Rev 5.0. Enqpy™ is a trademark of Enqpy™ Foundation Inc., licensed to NQP LLC. Full method and figures: enqpy.com/verify.